AF/
IFW

Attorney Docket No. 30014200-1057

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of	)	Group Art Unit 2167
	)	
Holtz *et al.*	)	Examiner: Mohammad Ali
	)	
Application No. 10/021,943	)	
	)	
Filed: December 12, 2001	)	
	)	
For: METHOD AND SYSTEM FOR COMPARING AND UPDATING FILE TREES	)	

Commissioner for Patents
Alexandria, VA 22313-1450

TRANSMITTAL OF APPELLANTS' BRIEF ON APPEAL

Dear Sir:

Appellants submit, in triplicate, Appellants' Brief on Appeal under 37 C.F.R. § 1.192 in support of the Notice of Appeal filed on June 30, 2005. Appellants also submit a check in the amount of $500 for the appeal brief fee as required by 37 C.F.R. § 41.20(b)(2).

The Commissioner is hereby authorized to credit overpayments or to charge any deficiency in a required fee to Deposit Account No. 19-3140. A duplicate copy of this sheet is enclosed.

Respectfully submitted,

Dated: August 29, 2005

By: _____
A. Wesley Ferrebee, Reg. No. 51,312

Customer No. 26263
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
8000 Sears Tower
Chicago, IL 60606-6404
(312) 876-8000

Attorney Docket No. 30014200-1057

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of	)	Group Art Unit 2167
	)	
Holtz *et al.*	)	Examiner: Mohammad Ali
	)	
Application No. 10/021,943	)	
	)	
Filed: December 12, 2001	)	
	)	
For: METHOD AND SYSTEM FOR COMPARING AND UPDATING FILE TREES	)	

Commissioner for Patents
Alexandria, VA 22313-1450

APPELLANTS' BRIEF ON APPEAL

Dear Sir:

In accordance with the provisions of 37 C.F.R. § 1.192, Appellants submit this Brief in support of the Appeal for the above-referenced application.

I. REAL PARTY IN INTEREST

The real party in interest in the present appeal is the Assignee, Sun Microsystems, a U.S. corporation. The Assignment was recorded in the U.S. Patent and Trademark Office.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals and no related interferences.

III. STATUS OF CLAIMS

Claims 1-24 are pending in this appeal, of which claims 1, 9, and 17 are independent. Claims 1-24 stand rejected and are now appealed. A listing of the claims appears in Appendix A.

IV. STATUS OF AMENDMENTS

A Request For Reconsideration After Final was filed on April 5, 2005, wherein no amendments of the claims were made. An Advisory Action was mailed April 22, 2005, stating that the Request For Reconsideration After Final had been considered but rejected because the Request did not place the application in condition for allowance.

V. SUMMARY OF INVENTION

Methods and systems in accordance with the present invention generally relate to a method for comparing two file structures and generating a sequence log of changes that will transform a first file structure into a second file structure. A file structure is represented as a tree of file folders, wherein each node may have child nodes that represent subfolders. The root node of the tree, indicating the top-most file folder, has no parent node. An exemplary file tree structure is illustrated in Figure 1 of the patent application.

The file comparing method in one embodiment in accordance with the present invention uses two file tree structures as inputs, performs a comparison of the file tree structures, and outputs a sequence log of file tree operations. The file tree operations are changes to the first file tree structure designed to transform the first file tree structure into the second file tree structure. See page 7, lines 10-17 and Figure 2 of the patent application.

In one embodiment in accordance with the present invention, the method recursively walks through the first file tree, comparing each folder's children with corresponding children in the second file tree. See page 7, lines 19-22 and Figure 4 of the patent application. In another embodiment in accordance with the present invention, the sequence log of file tree operations is optimized by transforming multiple file tree operations into a single file tree operation. See page 7, lines 24-27 and Figure 4 of the patent application.

Accordingly, methods and systems in accordance with the present invention presents an improved method, device, and computer product for producing a change log for updating a file tree.

VI. ISSUES

The issue on Appeal is whether claims 1-24 are properly rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 5,202,971 to *Henson et al.* (hereinafter "*Henson*") in view of U.S. Patent No. 5,842,213 to *Odom et al.* (hereinafter "*Odom*").

VII. GROUPING OF CLAIMS

The claims do not stand or fall together. Specific arguments as to the separate patentability of selected claims have been presented.

VIII. ARGUMENT

To render a claim unpatenable under 35 U.S.C. § 103(a), the Examiner bears the burden of establishing a *prima facie* case of obviousness. *In re Rinehart*, 531 F.2d 1048, 189 USPQ 143 (CCPA 1976); *In re Linter*, 458 F.2d 1013, 173 USPQ 560 (CCPA 1972). In order to establish a *prima facie* case of obviousness, all of the claim limitations must be taught or suggested by the

prior art. *See* MPEP 2143.03, *citing In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In the present application, the cited prior art fails to teach or suggest all of the limitations of any of pending claims 1-24. Thus, the Examiner has failed to established a *prima facie* case of obviousness for any of pending claims 1-24. Accordingly, the rejection of claims 1-24 should be reversed.

A. Claims 1, 9, and 17 Are Patentable Over the Combination of *Henson* and *Odom*

Neither *Henson*, nor *Odom*, nor the combination of *Henson* in view of *Odom* teaches or suggests all of the limitations of claim 1, 9, and 17. Namely, the references and combination thereof fail to teach or suggest comparing said first file structure to said second file structure, as recited in claims 1 and 17, or a comparator configured to compare said first file structure to said second file structure, as recited in claim 9. Moreover, the references and combinations thereof fail to teach or suggest generating one or more changes that transform said first file structure to said second file structure, or a comparator configured to generate one or more changes that transform said first file structure to said second file structure, as recited in claim 9.

1. "Comparing said first file structure to said second file structure"

The Examiner cites *Henson* as evidence that *Henson* in view of *Odom* teaches or suggests the comparing limitation found in claim 1. *Henson* states,

> With reference to Fig. 8, a client node may have access to files which reside in a remote server node. Such a client gains access to a server's files by mounting one of the server's directories. In the client node, the data structures created by a remote mount operation *compare* to those created by mounting a local entity in the following ways: Just as in the local case, a remote mount creates a vfs in the client node (e.g. block 54). Just as in the local case, use of a file in a virtual file system which contains remote files creates a vnode structure in the client node (e.g. block 57). (Col. 13, Lines 54-66; Emphasis added.)

The word "compare" as used in *Henson* does not mean "comparing," *i.e.*, the act of conducting a comparison, as is claimed in claim 1. In the context presented in *Henson*, the term "compare" is synonymous with the meaning "is consistent with," "to be worthy of comparison," and "to be regarded as similar or equal." Webster's Third New World International Dictionary of the English Language p. 462 (1993) (hereinafter "Webster's"). Webster's identifies the word "compare" in this context as an intransitive verb. Webster's also defines the meaning of the transitive verb form of the word "compare" as to mean, "to examine in order to discover similarities or differences." An intransitive verb cannot and does not take a direct object of a sentence while a transitive verb must take a direct object to complete the sentence. As an example, consider the verb "carried." An example of the use of the verb "carried" in its intransitive form is, "The sound of the choir "carried" through the cathedral." When "carried" is used in its transitive form, the same word projects a significantly different meaning. For example, "The truck carried the boxes." The same analysis can be applied to the term "compare."

Henson uses the word "compare" in its intransitive form to mean "to be worthy of comparison," or "to be regarded as similar or equal." Applicants use the word "compare" (the root of comparing) in its transitive form where "said first file structure to said second file structure" is the object of the verb. The meanings and use of the term "compare" are distinct. *Henson* does not communicate the same meaning of the term "compare" as Applicants, and thus does not teach or suggest the limitation of "comparing said first file structure to said second file structure." As *Odom* fails to correct this deficiency, *Henson* in view of *Odom* fails to establish a *prima facie* case of obviousness.

Moreover, *Henson* does not teach that the data structures created by a remote mount operation are or should be compared to those created by mounting a local entity, and thus does not teach the act of comparing a first and second file structure. *Henson* merely suggests that the data structures bear some similarities. The Examiner, however, has improperly inferred that *Henson* teaches that a comparison between data structures is actively performed or should be performed. Contrary to the Examiner's inference, *Henson* does not instruct one to make a comparison and offers no advantage or useful result for performing a comparison of the data structures, thus offering no suggestion to one of ordinary skill in the art to make such a comparison. As *Odom* again fails to correct this deficiency, *Henson* in view of *Odom* fails to establish a *prima facie* case of obviousness. For at least these reasons, the rejection of claim 1 should be reversed.

2. "Generating one or more changes that transform said first file structure to said second file structure"

With respect to the Examiner's argument that claim 1 is justly rejected since *Henson* in view of *Odom* teaches "transforming file structure," Applicants respectfully disagree because, as stated previously in the Response dated June 30, 2004, the pending claims do not possess the limitation of "transforming file structure." In the Office Action dated January 5, 2005 (p. 8), the Examiner states that Applicants argue that *Henson* and *Odom* do not teach 'transforming file structure,'" but that is not what Applicants argued. On the contrary, Applicants stated that "transforming file structure" is not in the pending claims. Whereas the Examiner is making a rejection on a limitation that is not in the claims, the rejection is improper. Accordingly, the rejection is without merit and should be withdrawn.

Nevertheless, *Henson* in view of *Odom* fails to teach or suggest "generating one or more changes that transform said first file structure to said second file structure." *Henson*, as cited by the Examiner, merely discloses updating a lock table and says nothing of generating a change to transform a first file structure to a second file structure (*See* Col. 6, Lines 56-61 of *Henson*). *Odom*, as cited by the Examiner, states:

> According to the present invention, this neutral form gives rise to greatly simplified modeling techniques, parallel processing in both data storage and retrieval operations, and simplified transfer of all or portions of the stored data. (Column 8, Lines 31-35).

As with *Henson*, *Odom* simply fails to disclose anything relating to generating one or more changes that transform a first file structure to a second file structure, as recited by claim 1. The cited text provides no support for the Examiner's assertions. Thus, the Examiner has failed to establish *prima facie* obviousness with respect to claim 1. By similar reason, the Examiner has failed to establish *prima facie* obviousness with respect to claims 9 and 17. Accordingly, the rejection of claims 1, 9, and 17 is erroneous and should be reversed.

B. Claims 2, 10, and 18 Are Patentable Over the Combination of *Henson* and *Odom*

The Examiner erroneously asserts that *Henson* teaches "recursively walking said first file structure." In support of this assertion, the Examiner points to Col. 26, Lines 65-67 *et seq.* of *Henson*. However, neither this nor any other part of *Henson* teaches or suggests recursively walking through a file structure. In fact, the entirety of *Henson* fails to include any form of the word "recursive." Moreover, *Odom* also lacks any occurrence of the word "recursive." Thus, the Examiner has failed to establish *prima facie* obviousness with respect to claim 2. By similar reasoning, the Examiner has also failed to establish *prima facie* obviousness with respect to claims 10 and 18. Accordingly, the rejection of claims 2, 10, and 18 are erroneous and should be withdrawn.

C. Claims 3, 11, and 19 Are Patentable Over the Combination of *Henson* and *Odom*

Applicants previously argued that the combination of *Henson* and *Odom* fails to teach or suggest that "changes comprise a sequence log of changes," as recited in claim 3. The Examiner responded in the final Official Action mailed January 5, 2005, asserting that the limitations of claim 3 are taught by *Henson* at Col. 2, Lines 40-42. Applicants note, however, that the cited portion of *Henson* reads: "Second, the user must have a separate "logon" on all the systems that are to be accessed" (Col. 2, Lines 40-42). The Examiner mistakenly applies the use of the word "log" as found in claim 3 to be equivalent to "logon" as found in *Henson.* The two words are distinct and possess decisively different meanings, as clearly evidenced by, *inter alia*, the *Microsoft Press Computer Dictionary*, Fifth Edition, pp. 316 and 318 (2002) (hereinafter "*Microsoft*"). A copy of those pages is provided with this Brief in Appendix B. On page 316 of *Microsoft*, "log" is defined as "a record of transactions or activities that take place on a computer system." By contrast, page 318 of *Microsoft* defines "logon" as "the process of identifying oneself to a computer after connecting to it over a communications line." Thus, *Henson* clearly fails to teach or suggest the "log" recited in claim 3.

Odom also fails to teach or suggest that the generated changes of claim 3 "comprise a sequence log of changes," as *Odom* is completely silent as to this limitation. Therefore the Examiner has failed to establish *prima facie* obviousness with respect to claim 3. By similar reasoning, the Examiner has also failed to establish *prima facie* obviousness with respect to claims 11 or 19. Accordingly, the rejection of claims 3, 11, and 19 is erroneous and should be reversed.

D. Claims 6, 14, and 22 Are Patentable Over the Combination of *Henson* and *Odom*

The Examiner erroneously asserts that *Henson* teaches "comparing one or more folders of said first file structure along with its children with a corresponding folder along with its children in said second file structure," as recited in claim 6. In support of this contention, that the Examiner points to *Henson*, which states:

> The two principal parameters to the mount operation are (1) the name of the device which holds the file to be mounted and (2) the path to the directory upon which the device's file tree is to be mounted. This directory must already be part of the node's file tree; i.e., it must be a directory in the root file system, or it must be a directory in a file system which has already been added (via a mount operation) to the node's file tree. (Col. 8, Lines 39-47)

The Examiner further offers another portion of *Henson*, which states:

> In the client node, the data structures created by a remote mount operation compare to those created by mounting a local entity in the following ways: Just as in the local case, a remote mount creates a vfs in the client node (e.g., block 54). Just as in the local case, use of a file in a virtual file system which contains remote files creates a vnode structure in the client node (e.g., block 57). Just as in the local case, the vnode structure has a pointer to a inode table entry (e.g., block 63). (Co. 13, Lines 57-60)

Clearly, neither of these passages makes any mention of folders of a file system, or children associated with folders of a file system. Furthermore, as previously explained, the occurrence of the word "compare" is in intransitive form, and thus does not mean the act of comparing as recited in claim 6. Moreover, *Odom* is also silent as to this limitation. Thus, the Examiner has failed to establish *prima facie* obviousness with respect to claim 6. By similar reasoning, the Examiner has also failed to establish *prima facie* obviousness with respect to claims 14 and 22. Accordingly, the rejection of claims 6, 14, and 22 is erroneous and should be reversed.

E. Claims 7, 15, and 23 Are Patentable Over the Combination of *Henson* and *Odom*

The Examiner erroneously asserts that *Henson* teaches "optimizing said sequenced log of changes," as recited in claim 7. In support of this contention, the Examiner cites *Henson*, which states:

> Utilizing the three modes to manage the use of the client cache optimizes overall system performance by combining both an overall average increase in read/write response speed with file integrity. (Col.19, Lines 1-4)

Clearly, *Henson* discloses that overall system performance is optimized and not a sequenced log of changes as recited in claim 7. *Henson* fails to teach or suggest the log of claim 3, and therefore also fails to teach or suggest optimizing this log as recited in claim 7. *Odom* is also silent as to the limitations of claim 7. Thus, the Examiner has failed to establish *prima facie* obviousness with respect to claim 7. By similar reasoning, the Examiner has also failed to establish *prima facie* obviousness with respect to claims 15 and 23. Accordingly, the rejection of claims 7, 15, and 23 is erroneous and should be reversed.

F. Claims 8, 16, and 24 Are Patentable Over the Combination of *Henson* and *Odom*

The Examiner erroneously asserts that *Henson* teaches "transforming a plurality of operations in said sequenced log of changes to a single operation," as recited in claim 8. In support of this contention the Examiner cites *Henson*, which states:

> This preserves the file system semantics.
>
> If the file is being accessed in a client node B, and the file is in ASYNC or READONLY mode, as shown in FIG. 13, the client operating system 11B does not convert the file descriptor and byte range within the file in the system call READ (file descriptor, N1) 16 to the device number and the logical block number in the device. The client does convert the file descriptor and byte range to a file handle, node identifier, and logical block number within the file. In the client cache 12B, there are blocks 17 that are designated by file handle, node identifier,

> and logical block number within the file. When a read 16 is issued from a client application 4B, the request for the read goes to the operating system 11B with the file descriptor and the byte range within the file. The operating system then looks in the client cache 12B. If the file handle, node identifier, and logical block number within the file is there, the cache 12B is read; on the other hand, if it is not there, the read is sent to the server. The server then takes the file handle and the logical block number within the file and converts it to a device number and logical block in the device. This conversion is necessary since the server cache 12A is managed by device number and block number within the device as it is in a standalone system. After the read is sent to the server, it is handled the same as if the read was coming from its own application in a standalone system as described with reference to FIG. 2. (Col. 19, Lines 34-60)

Despite the length of this passage, Applicants note that nowhere does *Henson* describe anything related to transforming a plurality of operations into a single operation. Furthermore, as previously explained, *Henson* fails to teach or suggest the log file of claims 3 and 7. Moreover, *Odom* is also silent as to the limitations of claim 8. Thus, the Examiner has failed to establish *prima facie* obviousness with respect to claim 8. By similar reasoning, the Examiner has also failed to establish *prima facie* obviousness with respect to claims 16 and 24. Accordingly, the rejection of claims 8, 16, and 24 is erroneous and should be reversed.

G. Claims 4-5, 12-13, and 20-21 Are Patentable Over the Combination of *Henson* and *Odom*

The rejection of claims 4-5, 12-13, and 20-21 relies on the assertion that claims 1, 9, and 17 are unpatentable. As previously elucidated, that assertion is erroneous. Accordingly, the rejection of claims 4-5, 12-13, and 20-21 is erroneous and should be reversed.

IX. Conclusion

Applicants respectfully submit that the outstanding rejections should be reversed, and that the application is in condition for allowance.

Respectfully submitted,

Dated: August 23, 2005

By:_____
A. Wesley Ferrebee, Reg. No. 51,312

Customer No. 26263
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
8000 Sears Tower
Chicago, IL 60606-6404
(312) 876-8000

APPENDIX A

Claims on Appeal

1. (Original) A method for comparing file tree descriptions comprising:

obtaining a first file structure;

obtaining a second file structure;

comparing said first file structure to said second file structure; and

generating one or more changes that transform said first file structure to said second file structure.

2. (Original) The method of claim 1 wherein said comparing further comprises:

recursively walking said first file structure.

3. (Original) The method of claim 1 wherein said changes comprise a sequence log of changes.

4. (Original) The method of claim 1 wherein said first file structure is a file tree index.

5. (Original) The method of claim 1 wherein said second file structure is a file tree index.

6. (Original) The method of claim 1 wherein said comparing further comprises:

comparing one or more folders of said first file structure along with its children with a corresponding folder along with its children in said second file structure.

7. (Original) The method of claim 3 further comprises:

optimizing said sequenced log of changes.

8. (Original) The method of claim 7 wherein said optimizing further comprising:

transforming a plurality of operations in said sequenced log of changes to a single operation.

9. (Original) A file tree comparator comprising:

a first file structure configured to be obtained;

a second file structure configured to be obtained;

a comparator configured to compare said first file structure to said second file structure; and

to generate one or more changes that transform said first file structure to said second file structure.

10. (Original) The file tree comparator of claim 9 wherein said step to compare further comprises:

to recursively walk said first file tree structure.

11. (Original) The file tree comparator of claim 9 wherein said changes comprise a sequence log of changes.

12. (Original) The file tree comparator of claim 9 wherein said first file structure is a file tree index.

13. (Original) The file tree comparator of claim 9 wherein said second file structure is a file tree index.

14. (Original) The file tree comparator of claim 9 wherein said step to compare further comprising:

to cause said file tree comparator to compare one or more folders of said first file structure along with its children with a corresponding folder along with its children in said second file structure.

15. (Original) The file tree comparator of claim 11 further comprising:

to optimize said sequenced log of changes.

16. (Original) The file tree comparator of claim 15 wherein said step optimize further comprising:

to transform a plurality of operations in said sequenced log of changes to a single operation.

17. (Original) A computer program product comprising:

a computer usable medium having computer readable program code embodied therein for comparing file tree descriptions, said computer program product comprising:

computer readable code configured to cause a computer to obtain a first file structure;

computer readable code configured to cause a computer to obtain a second file structure;

computer readable code configured to cause a computer to compare said first file structure to said second file structure; and

computer readable code configured to cause a computer to generate one or more changes that transform said first file structure to said second file structure.

18. (Original) The computer program product of claim 17 wherein said step to compare further comprising:

computer readable code configured to cause a computer to recursively walk said first file structure.

19. (Original) The computer program product of claim 17 wherein said changes comprise a sequence log of changes.

20. (Original) The computer program product of claim 17 wherein said first file structure is a file tree index.

21. (Original) The computer program product of claim 17 wherein said second file structure is a file tree index.

22. (Original) The computer program product of claim 17 wherein said step to compare further comprising:

computer readable code configured to compare one or more folders of said first file structure along with its children with a corresponding folder along with its children in said second file structure.

23. (Original) The computer program product of claim 18 further comprising:

computer readable code configured to optimize said sequenced log of changes.

24. (Original) The computer program product of claim 23 wherein said step to optimize further comprising:

computer readable code configured to transform a plurality of operations in said sequenced log of changes to a single operation.

IX. Conclusion

Applicants respectfully submit that the outstanding rejections should be reversed, and that the application is in condition for allowance.

Respectfully submitted,

Dated: August 29, 2005

By: _____
A. Wesley Ferrebee, Reg. No. 51,312

Customer No. 26263
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
8000 Sears Tower
Chicago, IL 60606-6404
(312) 876-8000

APPENDIX B

Microsoft Press Computer Dictionary, Fifth Edition, pp. 316 and 318 (2002)

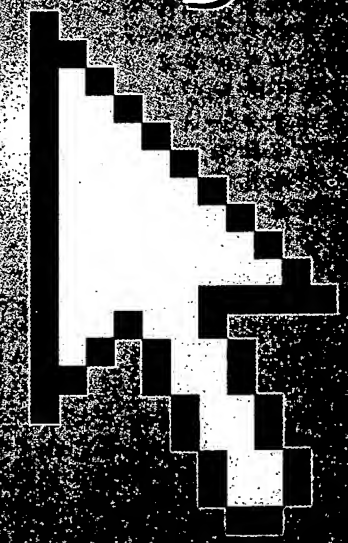
(see attached 4 pages)

Microsoft

OVER 10,000 ENTRIES

Microsoft® Computer Dictionary

Fifth Edition



- *Fully updated with the latest technologies, terms, and acronyms*
- *Easy to read, expertly illustrated*
- *Definitive coverage of hardware, software, the Internet, and more!*

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Computer Dictionary.--5th ed.
p. cm.
ISBN 0-7356-1495-4
1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.

AQ76.5. M52267 2002
004'.03--dc21 200219714

Printed and bound in the United States of America.

2 3 4 5 6 7 8 9 QWT 7 6 5 4 3 2

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *mspinput@microsoft.com.*

Acquisitions Editor: Alex Blanton
Project Editor: Sandra Haynes

Body Part No. X08-41929

identifier (LCID) identifies the primary language and any secondary language of a specific locale. *Acronym:* LCID. *See also* language identifier.

localhost *n.* The name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet. *See also* IP address, packet (definition 1), TCP/IP.

localization *n.* The process of altering a program so that it is appropriate for the geographic area in which it is to be used. Localization involves the customization or translation of the separated data and resources required for a specific region or language. For example, the developers of a word processing program must localize the sorting tables in the program for different countries or languages because the correct order of characters in one language might be incorrect in another. L10N is a common abbreviation for Localization, where the "L" in Localization is followed by 10 letters and ends with the letter "N."

localized version *n.* A version of a program that has been translated into another language. *Also called:* international version.

local loop *n.* The (end) portion of a telephone connection that runs from the subscriber to the local telephone exchange. *See also* last mile.

local memory *n.* In multiprocessor systems, the memory on the same card or high-speed bus as a particular processor. Typically, memory that is local to one processor cannot be accessed by another without some form of permission.

local newsgroups *n.* Newsgroups that are targeted toward a geographically limited area such as a city or educational institution. Posts to these newsgroups contain information that is specific to the area, concerning such topics as events, meetings, and sales. *See also* newsgroup.

local reboot *n.* A reboot of the machine that one is directly working on, rather than of a remote host. *See also* reboot.

LocalTalk *n.* An inexpensive cabling scheme used by AppleTalk networks to connect Apple Macintosh computers, printers, and other peripheral devices. *See also* AppleTalk.

local user profile *n.* A user profile that is created automatically on the computer the first time a user logs on to a computer. *See also* mandatory user profile, roaming user profile, user profile.

local variable *n.* A program variable whose scope is limited to a given block of code, usually a subroutine. *See also* scope (definition 1). *Compare* global variable.

location *n. See* address[1] (definition 1).

location-based service *n.* A service provided to a wireless mobile device based on the device's location. Location-based services can range from simple services, such as listing nearby restaurants, to more complex features, such as connecting to the Internet to monitor traffic conditions and find the least congested route to a destination.

lock *n.* **1.** A software security feature that requires a key or dongle in order for the application to run correctly. *See also* dongle. **2.** A mechanical device on some removable storage medium (for example, the write-protect notch on a floppy disk) that prevents the contents from being overwritten. *See also* write-protect notch.

locked file *n.* **1.** A file on which one or more of the usual types of manipulative operation cannot be performed—typically, one that cannot be altered by additions or deletions. **2.** A file that cannot be deleted or moved or whose name cannot be changed.

locked volume *n.* On the Apple Macintosh, a volume (storage device, such as a disk) that cannot be written to. The volume can be locked either physically or through software.

lockout *n.* The act of denying access to a given resource (file, memory location, I/O port), usually to ensure that only one program at a time uses that resource.

lock up *n.* A condition in which processing appears to be completely suspended and in which the program in control of the system will accept no input. *See also* crash[1].

log *n.* A record of transactions or activities that take place on a computer system. *See* logarithm.

logarithm *n.* Abbreviated log. In mathematics, the power to which a base must be raised to equal a given number. For example, for the base 10, the logarithm of 16 is (approximately) 1.2041 because $10^{1.2041}$ equals (approximately) 16. Both natural logarithms (to the base *e*, which is approximately 2.71828) and common logarithms (to the base 10) are used in programming. Languages such as C and Basic include functions for calculating natural logarithms.

logic array *n. See* gate array.

logic board *n.* Another name for motherboard or processor board. The term was used in conjunction with older computers to distinguish the video board (*analog board*) from the motherboard. *See also* motherboard.

logic bomb *n.* **1.** A logic error in a program that manifests itself only under certain conditions, usually when least expected or desired. The term *bomb* implies an error that causes the program to fail spectacularly. *See also* logic error. **2.** A type of Trojan horse that executes when certain conditions are met, such as when a user performs a specific action. **3.** *See* Year 2000 problem. **4.** *See* fork bomb.

logic chip *n.* An integrated circuit that processes information, as opposed to simply storing it. A logic chip is made up of logic circuits.

logic circuit *n.* An electronic circuit that processes information by performing a logical operation on it. A logic circuit is a combination of logic gates. It produces output based on the rules of logic it is designed to follow for the electrical signals it receives as input. *See also* gate (definition 1).

logic diagram *n.* A schematic that shows the connections between computer logic circuits and specifies the expected outputs resulting from a specific set of inputs.

logic error *n.* An error, such as a faulty algorithm, that causes a program to produce incorrect results but does not prevent the program from running. Consequently, a logic error is often very difficult to find. *See also* logic, semantics, syntax.

logic gate *n. See* gate (definition 1).

logic operation *n.* **1.** An expression that uses logical values and operators. **2.** A bit-level manipulation of binary values. *See also* Boolean operator.

logic programming *n.* A style of programming, best exemplified by Prolog, in which a program consists of facts and relationships from which the programming language is expected to draw conclusions. *See also* Prolog.

logic-seeking printer *n.* Any printer with built-in intelligence that lets it look ahead of the current print position and move the print head directly to the next area to be printed, thus saving time in printing pages that are filled with spaces.

logic symbol *n.* A symbol that represents a logical operator such as AND or OR. For example, the symbol + in Boolean algebra represents logical OR, as in A + B (read, "A or B," not "A plus B").

logic tree *n.* A logic specification method that uses a branching representation. Each of the tree's forks represents a decision point; the ends of the branches denote actions to be taken.

login *n. See* logon.

log in *vb. See* log on.

Logo *n.* A programming language with features that are heavily drawn from LISP. Logo is often used to teach programming to children and was developed originally by Seymour Papert at MIT in 1968. Logo is considered an educational language, although some firms have sought to make it more widely accepted in the programming community. *See also* LISP, turtle, turtle graphics.

logoff *n.* The process of terminating a session with a computer accessed through a communications line. *Also called:* logout.

log off *vb.* To terminate a session with a computer accessed through a communications line—usually a computer that is both distant and open to many users. *Also called:* log out. *Compare* log on.

logon *n.* The process of identifying oneself to a computer after connecting to it over a communications line. *Also called:* login.

log on *vb.* To gain access to a specific computer, a program, or a network by identifying oneself with a username and a password. *Also called:* log in. *Compare* log off.

logon script *n.* A file assigned to certain user accounts on a network system. A logon script runs automatically every time the user logs on. It can be used to configure a user's working environment at every logon, and it allows an administrator to influence a user's environment without managing all aspects of it. A logon script can be assigned to one or more user accounts. *Also called:* login script. *See also* user account.

logout *n. See* logoff.

log out *vb. See* log off.

LOL *n.* Acronym for laughing out loud. An interjection used in e-mail, online forums, and chat services to express